68491-US
TOK/mk

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: ２００３年 １月２７日

出 願 番 号
Application Number: 特願２００３－０１７６６８

[ST. 10/C]: [ＪＰ２００３－０１７６６８]

出 願 人
Applicant(s): 株式会社デンソー

２００３年１２月 １日

特許庁長官
Commissioner,
Japan Patent Office 今 井 康

出証番号 出証特２００３－３０９９０１０

【書類名】　特許願
【整理番号】　IP7507
【提出日】　平成15年 1月27日
【あて先】　特許庁長官殿
【国際特許分類】　G06F　9/44
　　F02D 43/04
【発明者】
　【住所又は居所】　愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内
　【氏名】　大井　正也
【発明者】
　【住所又は居所】　愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内
　【氏名】　植松　義貴
【発明者】
　【住所又は居所】　愛知県刈谷市昭和町１丁目１番地　株式会社デンソー内
　【氏名】　岩井　　明史
【特許出願人】
　【識別番号】　000004260
　【氏名又は名称】　株式会社デンソー
【代理人】
　【識別番号】　100100022
　【弁理士】
　【氏名又は名称】　伊藤　洋二
　【電話番号】　052-565-9911
【選任した代理人】
　【識別番号】　100108198
　【弁理士】
　【氏名又は名称】　三浦　高広
　【電話番号】　052-565-9911

【選任した代理人】

【識別番号】　100111578

【弁理士】

【氏名又は名称】　水野　史博

【電話番号】　052-565-9911

【手数料の表示】

【予納台帳番号】　038287

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　　明細書

【発明の名称】　コード生成装置、コード生成プログラム、シミュレーション装置、シミュレーションプログラム、モデル生成装置、およびモデル生成プログラム

【特許請求の範囲】

【請求項１】　モデルからソースコードを生成するコード生成装置であって、

自己の特定の部分を指定する部分指定子を含むモデルを取得するモデル取得手段と、

前記部分指定子によって指定された部分の取捨選択に関する選択情報を取得する選択情報取得手段と、

前記モデル取得手段が取得した前記モデルに含まれる前記部分指定子および前記選択情報取得手段が取得した前記選択情報に基づいて、前記モデルから前記部分指定子が指定する部分を取り除いたものからソースコードを生成する除去生成手段と、を備えたコード生成装置。

【請求項２】　前記部分指定子は、指定するモデルの部分を囲む部分指定ブロックから成り、

前記除去生成手段は、前記モデル取得手段が取得した前記モデルに含まれる前記部分指定ブロックおよび前記選択情報取得手段が取得した前記選択情報に基づいて、前記モデルから前記部分指定子が指定する部分を取り除いたものからソースコードを生成することを特徴とする請求項１に記載のコード生成装置。

【請求項３】　前記部分指定子は、指定する前記部分について記述される属性情報に含まれることを特徴とする請求項１に記載のコード生成装置。

【請求項４】　前記モデル取得手段が取得した前記モデルに含まれる前記部分指定子および前記選択情報取得手段が取得した前記選択情報との相関関係を示す相関情報を取得する相関情報取得手段を更に備え、

前記除去生成手段は、前記モデル取得手段が取得した前記モデルに含まれる前記部分指定子、前記選択情報取得手段が取得した前記選択情報、および前記相関情報取得手段が取得した前記相関情報に基づいて、前記モデルから前記部分指定

子が指定する部分を取り除いたものからソースコードを生成することを特徴とする請求項１に記載のコード生成装置。

【請求項５】　前記選択情報は、前記除去生成手段が生成するソースコードが対象とする機種の情報を含むことを特徴とする請求項１ないし４のいずれか１つに記載のコード生成装置。

【請求項６】　前記選択情報は、前記除去生成手段が生成するソースコードの仕向けの情報を含むことを特徴とする請求項１ないし５のいずれか１つに記載のコード生成装置。

【請求項７】　前記選択情報は、前記除去生成手段が生成するソースコードの使用目的の情報を含むことを特徴とする請求項１ないし６のいずれか１つに記載のコード生成装置。

【請求項８】　コンピュータを、

自己の特定の部分を指定する部分指定子を含むモデルを取得するモデル取得手段、

前記部分指定子によって指定された部分の取捨選択に関する選択情報を取得する選択情報取得手段、および

前記モデル取得手段が取得した前記モデルに含まれる前記部分指定子および前記選択情報取得手段が取得した前記選択情報に基づいて、前記モデルから前記部分指定子が指定する部分を取り除いたものからソースコードを生成する除去生成手段、として機能させるためのコード生成プログラム。

【請求項９】　自己の特定の部分を指定する部分指定子を含むモデルを取得するモデル取得手段と、

前記部分指定子によって指定された部分の取捨選択に関する選択情報を取得する選択情報取得手段と、

前記モデル取得手段が取得した前記モデルに含まれる前記部分指定子および前記選択情報取得手段が取得した前記選択情報に基づいて、前記モデルから前記部分指定子が指定する部分を取り除いたものに記述された機能を実行する除去実行手段と、を備えたシミュレーション装置。

【請求項１０】　コンピュータを、

自己の特定の部分を指定する部分指定子を含むモデルを取得するモデル取得手段、

前記部分指定子によって指定された部分の取捨選択に関する選択情報を取得する選択情報取得手段、および

前記モデル取得手段が取得した前記モデルに含まれる前記部分指定子および前記選択情報取得手段が取得した前記選択情報に基づいて、前記モデルから前記部分指定子が指定する部分を取り除いたものに記述された機能を実行する除去実行手段、として機能させるためのシミュレーションプログラム。

【請求項１１】　自己の特定の部分を指定する部分指定子を含むモデルを取得するモデル取得手段と、

前記部分指定子によって指定された部分の取捨選択に関する選択情報を取得する選択情報取得手段と、

前記モデル取得手段が取得した前記モデルに含まれる前記部分指定子および前記選択情報取得手段が取得した前記選択情報に基づいて、前記モデルから前記部分指定子が指定する部分を取り除いたモデルを生成する除去生成手段と、を備えたモデル生成装置。

【請求項１２】　コンピュータを、

自己の特定の部分を指定する部分指定子を含むモデルを取得するモデル取得手段、

前記部分指定子によって指定された部分の取捨選択に関する選択情報を取得する選択情報取得手段、および

前記モデル取得手段が取得した前記モデルに含まれる前記部分指定子および前記選択情報取得手段が取得した前記選択情報に基づいて、前記モデルから前記部分指定子が指定する部分を取り除いたモデルを生成する除去生成手段、として機能させるためのモデル生成プログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、モデルによるプログラムの開発において用いられるコード生成装置

、コード生成プログラム、シミュレーション装置、シミュレーションプログラム、モデル生成装置、およびモデル生成プログラムに関するものである。

【０００２】

【従来の技術】

従来、例えば車両のエンジンＥＣＵの作動のためのプログラムにおいては、開発者が直接そのプログラムのソースコードを記述せず、目的とするプログラムの機能を、より作成が簡易で視認性の良い「モデル」という形態で記述する場合がある。開発者は、このモデルに対応したプログラム開発環境がインストールされたワークステーション、パーソナルコンピュータ等を用い、そのモデルをもとに目的とするプログラムの動作の確認および試験を行い、更にそのモデルからソースコードを生成する。目的とするプログラムの動作の確認および試験の機能を担うプログラムをシミュレーションツールと呼び、モデルからソースコードを生成する機能を担うプログラムをコード生成ツールと呼ぶ。コード生成ツールやシミュレーションツールは、最初からモデル開発環境に統合されている場合もあれば、追加モジュールとしてモデル開発環境に後から組み込まれる場合もある。

【０００３】

モデルに対応したプログラム開発環境としては、Ｍａｔｌａｂ（登録商標）がある。Ｍａｔｌａｂ（登録商標）においては、開発者はＭａｔｌａｂ（登録商標）の一機能であるＳｉｍｕｌｉｎｋ（登録商標）を用いて、目的とするプログラムの機能を、ブロックと呼ばれる機能単位の組み合わせとして記述する。この組み合わされたブロックの集合体がモデルである。Ｓｉｍｕｌｉｎｋ（登録商標）のブロックとしては、例えばサイン関数を生成するブロック、ファイルからデータを読み出すブロック、入力されたデータに対する特定の四則演算を行うブロック、ブロックの組み合わせから成る上位のブロックとしてのサブシステム等がある。

【０００４】

このように、プログラム開発において、開発者がモデルを生成し、コード生成ツールによってモデルからソースコードが生成され、最終的にソースコードからプログラムが生成されるという手順を取ると、当初のモデルによってプログラム

が表現されることで当該プログラムの機能の視認性が良くなる等の有益な効果を得ることができる。

【０００５】

また、上記したエンジンＥＣＵのためのプログラムにおいては、Ｖ６（Ｖ型６気筒）、Ｖ８（Ｖ型８気筒）、Ｉ６（直列６気筒）等のエンジンの機種別、日本向け、欧州向け、米国向け等の仕向け別、あるいはメーカー納入用途、デバッグ用途等の目的別といったバリエーションに応じて複数の種類のプログラムを作成する場合がある。

【０００６】

これらのバリエーションは、互いに共通する部分を多く有するので、バリエーション毎に別個にモデルの開発を行うよりは、各バリエーションを含む１つのモデルを作成する方が、プログラムの開発、管理労力の低減に繋がる。

【０００７】

１つのモデルからバリエーションに応じて複数の種類のソースコードを作成するためには、モデルにおいてバリエーションの違いが表現され、どのバリエーションのソースコードを生成するかを切り替えることができるようになっている必要がある。これを実現する方法としては、従来、スイッチ機能を有するブロックを用いた切り替え方法が用いられている。図１７に、この従来の切り替え方法で表現したモデルの一部分を示す。

【０００８】

Ｖ６用サブシステム５１、Ｖ８用サブシステム５２、Ｉ６用サブシステム５３は、ブロックの集合体であり、それぞれがＶ６エンジン用、Ｖ８エンジン用、Ｉ６エンジン用のバリエーションに対応した処理が記述されている。これらのサブシステムは、それぞれが有するブロックによって規定された所定の演算結果を出力するようになっている。入力端子５５は、選択ブロック５４に対して設定可能な値としての選択信号ａを入力する端子である。

【０００９】

選択ブロック５４は、入力端子５５からの入力値に基づいて、Ｖ６用サブシステム５１、Ｖ８用サブシステム５２、Ｉ６用サブシステム５３から入力される演

算結果のいずれか１つを出力する機能を有するブロックである。

【００１０】

このような部分を有するモデルが、コード生成ツールによってソースコードに変換された場合、そのソースコードのうち図１７に示した部分に対応する部分は、例えば図１８の様になる。

【００１１】

この図のソースコード中のＳｗｉｔｃｈ文は、中括弧 {} で囲まれた領域において、引数ａの値が１、２、または３の場合、それぞれｃａｓｅ　１：、ｃａｓｅ　２：、ｃａｓｅ　３：と直後のｂｒｅａｋ文との間にある文を実行することを規定する文である。Ｓｗｉｔｃｈ文が選択ブロック５４に対応し、引数ａが選択信号ａに対応し、ｃａｓｅ　１：から直後のｂｒｅａｋ文までの間のコードがＶ６用サブシステム５１に対応し、ｃａｓｅ　２：から直後のｂｒｅａｋ文までの間のコードがＶ８用サブシステム５２に対応し、ｃａｓｅ　３：から直後のｂｒｅａｋ文までの間のコードがＩ６用サブシステム５３に対応する。

【００１２】

【発明が解決しようとする課題】

このように生成されたソースコードは、例えば設定された選択信号ａの値が１であればＶ６用サブシステム５１に対応するソースコードの部分が実行されるようになり、Ｖ８、Ｉ６に対応するソースコードの部分は実行されることがない。したがって、生成されたコードは選択信号ａによってそれぞれのバリエーションに対応することができるが、使用することのないソースコードの部分まで形式的に含んでいるので、このソースコードがコンパイルされると、無駄なソースコード部分を含むプログラムが生成され、プログラムサイズが増大してしまう。この結果、プログラムを格納するメモリの容量が圧迫されてしまう。

【００１３】

本発明は上記点に鑑みて、複数のバリエーションに対応するモデルからソースコードが生成されるプログラム開発環境において、生成されるソースコードから当初のモデルの不要な部分に対応するソースコードが除かれているようにすることを目的とする。

【００１４】

またプログラム開発環境においては、モデルからソースコードが生成されるまでの間の段階として中間コードが生成される場合もあることに鑑みて、複数のバリエーションに対応するモデルからさらにモデルが生成されるプログラム開発環境において、生成されるモデルから当該モデルの不要な部分に対応するモデルが除かれているようにすることも目的とする。

【００１５】

また、上記のようなプログラム開発環境において、モデルからオブジェクトコードを生成する前に、あらかじめ当該オブジェクトコードの動作を確認するために、当該モデルを直接実行するシミュレーションプログラムまたは装置が提供されることがある。このようなシミュレーションプログラム等は、モデル生成されるオブジェクトコードと同等の動作を実現する必要があることに鑑み、複数のバリエーションに対応するモデルを実行して動作を確認するプログラム開発環境において、実行されるモデルを、当初のモデルの不要な部分に対応する機能を除いた形で実行することも目的とする。

【００１６】

【課題を解決するための手段】

上記した課題を解決するための請求項１に記載の発明は、モデルからソースコードを生成するコード生成装置であって、自己の特定の部分を指定する部分指定子を含むモデルを取得するモデル取得手段と、前記部分指定子によって指定された部分の取捨選択に関する選択情報を取得する選択情報取得手段と、前記モデル取得手段が取得した前記モデルに含まれる前記部分指定子および前記選択情報取得手段が取得した前記選択情報に基づいて、前記モデルから前記部分指定子が指定する部分を取り除いたものからソースコードを生成する除去生成手段と、を備えたコード生成装置である。

【００１７】

これによって、除去生成手段が、モデル取得手段が取得したモデルに含まれる部分指定子および選択情報取得手段が取得した選択情報に基づいて、モデルから部分指定子が指定する部分を取り除いたものからソースコードを生成するので、

部分指定子で指定された、不要となり得るモデルの部分は、選択情報によっては生成されるソースコードから取り除かれる。したがって、複数のバリエーションに対応するモデルからソースコードが生成されるプログラム開発環境において、生成されるソースコードから当該モデルの不要な部分に対応するソースコードが除かれているようにすることが可能となる。

【００１８】

なお、部分指定子は１つのモデル中に複数含まれていても構わない。また、除去生成手段が「モデルから部分指定子が指定する部分を取り除いたもの」とは、モデルに含まれる部分指定子のうちの１つまたは複数が指定する部分を取り除いたもの、およびモデルに含まれる「全ての」部分指定子が指定する部分を取り除いたものを含む概念である。

【００１９】

また、請求項２に記載の発明は、請求項１に記載のコード生成装置において、前記部分指定子は、指定するモデルの部分を囲む部分指定ブロックから成り、前記除去生成手段は、前記モデル取得手段が取得した前記モデルに含まれる前記部分指定ブロックおよび前記選択情報取得手段が取得した前記選択情報に基づいて、前記モデルから前記部分指定子が指定する部分を取り除いたものからソースコードを生成することを特徴とする。

【００２０】

また、請求項３に記載の発明は、請求項１に記載のコード生成装置において、前記部分指定子は、指定する前記部分について記述される属性情報に含まれることを特徴とする。

【００２１】

また、請求項４に記載の発明は、請求項１に記載のコード生成装置において、前記モデル取得手段が取得した前記モデルに含まれる前記部分指定子および前記選択情報取得手段が取得した前記選択情報との相関関係を示す相関情報を取得する相関情報取得手段を更に備え、前記除去生成手段は、前記モデル取得手段が取得した前記モデルに含まれる前記部分指定子、前記選択情報取得手段が取得した前記選択情報、および前記相関情報取得手段が取得した前記相関情報に基づいて

、前記モデルから前記部分指定子が指定する部分を取り除いたものからソースコードを生成することを特徴とする。

【００２２】

また、請求項５に記載の発明は、請求項１ないし４のいずれか１つに記載のコード生成装置において、前記選択情報は、前記除去生成手段が生成するソースコードが対象とする機種の情報を含むことを特徴とする。

【００２３】

また、請求項６に記載の発明は、請求項１ないし５のいずれか１つに記載のコード生成装置において、前記選択情報は、前記除去生成手段が生成するソースコードの仕向けの情報を含むことを特徴とする。

【００２４】

また、請求項７に記載の発明は、請求項１ないし６のいずれか１つに記載のコード生成装置において、前記選択情報は、前記除去生成手段が生成するソースコードの使用目的の情報を含むことを特徴とする。

【００２５】

また、請求項８に記載の発明は、請求項１に記載の発明と同じ機能をプログラムとして実現するものである。

【００２６】

また、請求項９に記載の発明は、自己の特定の部分を指定する部分指定子を含むモデルを取得するモデル取得手段と、前記部分指定子によって指定された部分の取捨選択に関する選択情報を取得する選択情報取得手段と、前記モデル取得手段が取得した前記モデルに含まれる前記部分指定子および前記選択情報取得手段が取得した前記選択情報に基づいて、前記モデルから前記部分指定子が指定する部分を取り除いたものに記述された機能を実行する除去実行手段と、を備えたシミュレーション装置である。

【００２７】

これによって、除去実行手段が、モデル取得手段が取得したモデルに含まれる部分指定子および選択情報取得手段が取得した選択情報に基づいて、モデルから部分指定子が指定する部分を取り除いたものに記述された機能を実行するので、

部分指定子で指定された、不要となり得るモデルの部分は、選択情報によっては生成されるモデルから取り除かれる。

【００２８】

したがって、複数のバリエーションに対応するモデルを実行して動作を確認するプログラム開発環境において、実行されるモデルから当該モデルの不要な部分に対応するソースコードが除かれているようにすることが可能となる。

【００２９】

また、請求項１０に記載の発明は、請求項９に記載の発明と同じ機能をプログラムとして実現するものである。

【００３０】

また、請求項１１に記載の発明は、自己の特定の部分を指定する部分指定子を含むモデルを取得するモデル取得手段と、前記部分指定子によって指定された部分の取捨選択に関する選択情報を取得する選択情報取得手段と、前記モデル取得手段が取得した前記モデルに含まれる前記部分指定子および前記選択情報取得手段が取得した前記選択情報に基づいて、前記モデルから前記部分指定子が指定する部分を取り除いたモデルを生成する除去生成手段と、を備えたモデル生成装置である。

【００３１】

これによって、除去生成手段が、モデル取得手段が取得したモデルに含まれる部分指定子および選択情報取得手段が取得した選択情報に基づいて、モデルから部分指定子が指定する部分を取り除いたモデルを生成するので、部分指定子で指定された、不要となり得るモデルの部分は、選択情報によっては生成されるモデルから取り除かれる。したがって、複数のバリエーションに対応するモデルを実行して動作を確認するプログラム開発環境において、実行されるモデルを、当該モデルの不要な部分に対応する機能を除いた形で実行することが可能となる。

【００３２】

また、請求項１２に記載の発明は、請求項１１に記載の発明と同じ機能をプログラムとして実現するものである。

【００３３】

【発明の実施の形態】

（第１実施形態）

図１に、本発明の第１実施形態に係る、コード生成装置としてのパーソナルコンピュータ１の構成を示す。パーソナルコンピュータ１は、ディスプレイ１１、入力装置１２、ＲＡＭ１３、ＲＯＭ１４、ＨＤＤ１５、およびＣＰＵ１６から構成される。

【００３４】

ディスプレイ１１は、ＣＰＵ１６から入力された映像信号を、ユーザに対して映像として表示する。

【００３５】

入力装置１２は、キーボード、マウス等から構成され、ユーザが操作することにより、その操作に応じた信号をＣＰＵ１６に出力する。

【００３６】

ＣＰＵ１６は、パーソナルコンピュータ１に電源が投入されることによって起動すると、ＲＯＭ（リードオンリーメモリー）１４から所定のブートプログラムを読み出して実行し、このブートプログラムに規定されるＭＳ－Ｗｉｎｄｏｗｓ（登録商標）等のオペレーティングシステム（以下ＯＳと記す）その他のプログラムをＨＤＤ（ハードディスクドライブ）１５から読み出して実行することにより、起動処理を行う。起動処理以後電源が遮断されるまで、ＣＰＵ１６は、入力装置１２からの信号、ＯＳによって予め定められたスケジュール等に基づいて、ＨＤＤ１５に記録されている各種プログラムを当該ＯＳ上のプロセスとして実行する。また、上記した起動処理およびプロセスにおいて、ＣＰＵ１６は必要に応じて入力装置１２から信号の入力を受け付け、またディスプレイ１１に映像信号を出力し、またＲＡＭ１３、ＨＤＤ１５に対してデータの読み出し／書き込みの制御を行う。

【００３７】

図２に、ＨＤＤ１５に保存され、ＯＳ上のプロセスとして実行されるプログラムの１つであるコード生成ツール２の構成および作動の概念図を示す。コード生成ツール２は、ＯＳの仕様に従ったユーザの入力装置１２の操作によって起動し

、その後ユーザの入力装置１２の操作に基づいて、モデルからソースコードを生成するプログラムである。

【００３８】

ソースコードとは、例えばＣ＋＋等のプログラミング言語の仕様に則ってプログラム開発者が作成するプログラムの一表現形態である。ソースコードはコンパイラ、リンカ等に入力されることで、ＣＰＵ等が直接実行できるオブジェクトコードに変換される。なお、オブジェクトコードもプログラムの一表現形態である。

【００３９】

モデルとは、ソースコードよりも記述が簡易になり、かつ人による可読性が高くなる目的で定められたモデル言語仕様に基づいて作成されるプログラムの一表現形態である。モデルとしては、例えばＳｉｍｕｌｉｎｋ（登録商標）によって作成されるＳｉｍｕｌｉｎｋモデルがある。Ｓｉｍｕｌｉｎｋ（登録商標）モデルが、Ｍａｔｌａｂ（登録商標）上で動作するＲｅａｌ　Ｔｉｍｅ　Ｗｏｒｋｓｈｏｐ（登録商標：以下ＲＴＷと記す）に入力されることにより、このＲＴＷが当該Ｓｉｍｕｌｉｎｋ（登録商標）モデルに対応するソースコードを生成する。

【００４０】

Ｓｉｍｕｌｉｎｋ（登録商標）モデルは、ブロックと呼ばれる機能単位の組み合わせからなる集合体として作成される。組み合わせとは、機能単位間の入出力の繋がりをいう。ブロックの具体例としては、例えば入力される２つの数値データを加算して出力する加算ブロック等がある。また、複数のブロックの集合体をサブシステムというブロックの一種として定義することもできる。すなわち、ブロックの中にブロックが含まれるような入れ子構造のブロックの作成も許される。また、Ｓｉｍｕｌｉｎｋ（登録商標）モデル中の各ブロックは、それぞれ属性情報を有することができる。属性情報としては、例えば当該ブロックの名称、入出力するデータの値域定義等がある。

【００４１】

本実施形態におけるモデルも、Ｓｉｍｕｌｉｎｋ（登録商標）モデルと同様にブロックから成り、ブロックの入れ子構造が許され、各ブロックが属性情報を有

することができる。

【００４２】

コード生成ツール２は、その機能面から生成モデル抽出エンジン２１、コード生成エンジン２２、生成ルール２３に分けることができる。

【００４３】

生成モデル抽出エンジン２１は、入力されるモデル３および選択情報４に基づいて中間モデルを生成する。この中間モデルは、モデルの一種である。

【００４４】

コード生成エンジン２２は、生成モデル抽出エンジン２１によって生成された中間モデルからソースコードを生成する。この中間モデルからソースコードを生成するための規則を定めた情報が生成ルール２３である。生成ルールはコード生成ツール２の一部として存在していてもよいし、あるいはコード生成ツール２の外部のファイルとしてＨＤＤ１５に保存されていてもよい。なお、生成モデル抽出エンジン２１とコード生成エンジン２２との区別は、あくまでもコード生成ツール２の機能面から見た区別であって、必ずしもプログラムとしては分離している必要はない。実際、本実施形態においては、生成モデル抽出エンジン２１とコード生成エンジン２２とは一体のプログラムとして実現される。

【００４５】

生成されたソースコードは、ＨＤＤ１５に記録されているコンパイラ、リンカ５を実行することによってオブジェクトコードに変換される。

【００４６】

図３に、コード生成ツール２に入力されるモデル３の一例を示す。このモデルは、車両のエンジンＥＣＵに搭載されるプログラムのためのモデルの一部であり、Ｖ６スタート３１、Ｖ６用サブシステム３２、Ｖ６エンド３３、Ｖ８スタート３４、Ｖ８用サブシステム３５、Ｖ８エンド３６、Ｉ６スタート３７、Ｉ６用サブシステム３８、およびＩ６エンド３９から構成される。

【００４７】

Ｖ６スタート３１、Ｖ６エンド３３、Ｖ８スタート３４、Ｖ８エンド３６、Ｉ６スタート３７、およびＩ６エンド３９は部分指定ブロックである。部分指定ブ

ロックは上述したブロックの一種であり、本発明における部分指定子に相当する。さらに部分指定ブロックは、Ｖ６スタート３１、Ｖ８スタート３４、Ｉ６スタート３７のスタートブロック、およびＶ６エンド３３、Ｖ８エンド３６、Ｉ６エンド３９のエンドブロックに分類される。１種類のスタートブロックには必ず対応する同種のエンドブロックが存在する。図３においては、Ｖ６スタート３１とＶ６エンド３３、Ｖ８スタート３４とＶ８エンド３６、およびＩ６スタート３７とＩ６エンド３９がそれぞれ対応するブロックである。

【００４８】

Ｖ６用サブシステム３２、Ｖ８用サブシステム３５、およびＩ６用サブシステム３８は、ブロックの集合体を有するブロック、すなわち入れ子構造のブロックである。それぞれのサブシステムは、Ｖ６エンジン、Ｖ８エンジン、Ｉ６エンジンといった、対象とする機種の制御に特有な機能を記述するブロックから構成されている。

【００４９】

なお、上記したそれぞれのブロックは、Ｖ６スタート、Ｉ６サブシステム等の自己の名称を属性情報として有している。また、スタートブロックは自己がスタートブロックである旨の情報を属性情報に有し、エンドブロックは自己がエンドブロックである旨の情報を属性情報に有している。

【００５０】

部分指定ブロックは、互いに対応する１対のスタートブロックとエンドブロックとの組としてモデル３中に現れる。そして、ブロック中のスタートブロックと、それに対応するエンドブロックとに前後を囲まれるブロック、サブシステム等は、当該部分指定ブロックによって、Ｖ６用、Ｖ８用、Ｉ６用等のバリエーションに対応する部分であると指定されることになる。

【００５１】

図４に、各サブシステムにおいてブロックの記述が異なる例として、入力されるエンジンの各気筒の状態を加算することでエンジン全体の状態を出力する機能を表現するブロックを示す。図４（ａ）がＶ６およびＩ６のサブシステム用のブロックであり、図４（ｂ）がＶ８のサブシステム用のブロックである。Ｖ６、Ｉ

６のエンジンは気筒数が６であるので、６本の入力端子を有するブロックが用いられる。また、Ｖ８のエンジンは気筒数が８であるので、８本の入力端子を有するブロックが用いられる。

【００５２】

図５に、コード生成ツール２において、ユーザが所定の操作をすることによって開始される、モデルからソースコードを生成する処理を示す。所定の操作とは、ユーザがコード生成ツール２に入力するモデル３、選択情報４を指定するために入力装置１２を操作し、さらに図５の処理を開始するよう要求するために入力装置１２を操作することをいう。なお、本実施形態においては、選択情報４は特定のスタートブロックの名称である。また、選択情報４は、ＨＤＤ１５中の特定のファイル中に予め設定されていてもよいし、ユーザが選択情報４の指定時に直接入力して設定してもよい。以下、図５の処理について説明する。

【００５３】

まずステップ５１０で、ユーザによって指定されたモデル３を読み込む。具体的には、モデル３として保存されているＨＤＤ１５中のファイルを読み出してＲＡＭ１３に書き込む。

【００５４】

次にステップ５２０で、ユーザによって指定された選択情報４を読み込む。具体的には、選択情報４として保存されているＨＤＤ１５中のファイルを読み出してＲＡＭ１３に書き込む。あるいは、入力装置１２から直接設定された情報を選択情報４としてＲＡＭ１３に書き込む。

【００５５】

次にステップ５３０では、モデル３中のスタートブロックの検索を行う。具体的には、ＲＡＭ１３に書き込んだモデル３に含まれる全てのブロックについて、それがスタートブロックであるか否かを判定し、スタートブロックであると判定したもの全てについて、そのブロックの属性情報、モデル３内の位置等の情報をスタートブロックリストとしてＲＡＭ１３に書き込む。

【００５６】

次にステップ５３５で、スタートブロックリストの先頭にあるスタートブロッ

クの情報を読み出し、その後読み出したスタートブロックの情報をスタートブロックリストから削除する。先頭のスタートブロックの情報が削除されることで、次のスタートブロックの情報が先頭のスタートブロックの情報となる。

【００５７】

次にステップ５４０で、当該スタートブロックの名称がＲＡＭ１３に記録された選択情報４と一致しているか否かを判定する。一致していると判定すると、処理はステップ５５５に進む。一致していないと判定すると、処理はステップ５５０に進む。

【００５８】

ステップ５５０では、当該スタートブロックおよびそれと対になるエンドブロックに囲まれた部分、すなわち当該部分指定子によって指定される部分を削除する。具体的には、当該スタートブロックの位置から、モデル３における後段方向へ向けて、当該スタートブロックと対になるエンドブロックを検索する。そしてＲＡＭ１３中のモデル３を、見つかったエンドブロックから当該スタートブロックまでのブロックが削除されたモデル３に書き換える。そして処理はステップ５５５に進む。

【００５９】

ステップ５５５では、当該スタートブロックおよびこれと対になるエンドブロックを、ステップ５５０で書き換えられたモデル３から削除する。すなわち、ＲＡＭ１３中のモデル３を、そのモデル３から当該スタートブロックおよびエンドブロックが削除されたモデル３に書き換える。

【００６０】

次にステップ５６０では、当該スタートブロックが最後のスタートブロックであるか否かを判定する。具体的には、スタートブロックリストにスタートブロックの情報が無ければ肯定であると判定し、あれば否定であると判定する。否定と判定すれば、処理はステップ５３５に戻る。肯定の場合、処理はステップ５７０に進み、ＲＡＭ１３に記録されているモデル３から、生成ルール２３に従ってソースコードを生成し、生成されたソースコードをＨＤＤ１５中にファイルとして書き込む。そして処理は終了する。

【００６１】

このような作動によって、例えば図３で示されたブロックを含むモデル３と、「Ｖ６」を示す選択情報４を入力してコード生成ツール２を作動させると、コード生成ツール２のステップ５１０～５６０の処理によって、図６に示すような中間モデルが生成される。そしてステップ５７０によってこの中間モデルに対応するソースコードが生成ルール２３に従って生成される。したがって、この生成されたソースコードにはＶ８用、Ｉ６用に特化されたコードは含まれることはない。

【００６２】

なお、ステップ５１０～５６０の処理は、図２に示した生成モデル抽出エンジン２１の機能を実現し、ステップ５７０の処理は、コード生成エンジン２２の機能を実現する。

【００６３】

以上のような構成および作動のパーソナルコンピュータ１、コード生成ツール２の効果について以下説明する。

【００６４】

Ｖ６、Ｖ８、Ｉ６等、複数のバリエーションに対応するモデルを作成し、ソースコードの作成時に個々のバリエーションにのみ向けられたソースコードを作成したい場合、それぞれのバリエーションに対応する部分を、そのバリエーションに固有の名称を持つ部分指定ブロックで囲んだモデル３を作成する。そして、ソースコード作成時に、選択情報４として特定のバリエーションの名称をファイルまたはユーザによる操作によってコード生成ツール２に入力し、また当該モデル３をコード生成ツール２に入力する。

【００６５】

これによって、コード生成ツール２は選択情報４に示された名称以外の名称の部分指定ブロックに囲まれたサブシステム等をモデル３から削除した上でソースコードを生成する。したがって、複数のバリエーションに対応するモデルからソースコードが生成される開発環境において、当該ソースコードから不要なバリエーションに対応するソースコードが除かれていることになる。ひいては、このソ

ースコードからコンパイル、リンクによって生成されるプログラムサイズの低減、およびこのプログラムを記録するメモリの容量の圧迫の緩和が実現される。

【００６６】

また、複数のバリエーションに対応するモデルからさらにモデルが生成されるプログラム開発環境において、生成されるモデルから当初のモデルの不要な部分に対応するモデルが除かれているようにすることも可能となる。

【００６７】

（第２実施形態）

以下、本発明の第２実施形態について説明する。本実施形態も、図１および図２で示したようなハードウェア構成およびソフトウェア構成から成っている。なお、本実施形態において第１実施形態と同一の部分があれば、その説明は省略または簡略化する。

【００６８】

本実施形態が第１実施形態と異なるのは、第１実施形態の部分指定子が部分指定ブロックであったことに対し、本実施形態の部分指定子がサブシステムの属性情報に含まれることである。

【００６９】

図７に、本実施形態においてコード生成ツール２に入力されるモデル３の一部を例示する。このモデル３は、Ａサブシステム７１、Ｂサブシステム７２、Ｃサブシステム７３、Ｄサブシステム７４、およびＥサブシステム７５を有している。また、上記したサブシステムは、それぞれ属性情報７１ａ、７２ａ、７３ａ、７４ａ、７５ａを有している。各属性情報は、部分指定子としての選択情報の項目を有し、その項目の値として、属性情報７１ａにはＶ６が、属性情報７２ａにはＶ８が、属性情報７３ａにはＩ６が、属性情報７４ａにはＶ６およびＶ８が、属性情報７５ａにはＩ６が、設定されている。

【００７０】

図８に、本実施形態のコード生成ツール２において、ユーザが所定の操作をすることによって開始される、モデルからソースコードを生成する処理を示す。

【００７１】

ステップ８１０、８２０の処理は図３に示したステップ５１０、５２０の処理と同等である。ただし、本実施形態においては、選択情報４は、Ｖ６、Ｖ８、Ｉ６等の選択情報の値である。

【００７２】

次にステップ８２５では、モデル３中のサブシステムの検索を行う。具体的には、ＲＡＭ１３に書き込んだモデル３に含まれる全てのブロックについて、サブシステムであるか否かを判定し、サブシステムであると判定したもの全てについて、そのサブシステムの属性情報、モデル３内の位置等の情報をサブシステムリストとしてＲＡＭ１３に書き込む。

【００７３】

次にステップ８３０で、サブシステムリストの先頭にあるサブシステムの属性情報等の情報を読み出し、その後当該サブシステムの情報をサブシステムリストから削除する。そしてステップ８４０で、読み出した属性情報中の使用バリエーションの項目に、選択情報４と一致するものがあるか否かを判定する。一致するものが無ければ処理はステップ８５０に進み、ＲＡＭ１３中のモデル３を、モデル３から当該サブシステムが削除されたものに書き換え、その後ステップ８６０に進む。ステップ８４０で一致するものがあれば、処理は直接ステップ８６０に進む。

【００７４】

ステップ８６０では、全てのサブシステムの属性情報について読み込みが完了したか否かを判定する。具体的には、サブシステムリストにサブシステムの情報が無ければ肯定であると判定し、あれば否定であると判定する。否定と判定すれば、処理はステップ８３０に戻る。肯定の場合、処理はステップ８７０に進み、ＲＡＭ１３に記録されているモデル３から、生成ルール２３に従ってソースコードを生成し、生成されたソースコードをＨＤＤ１５中にファイルとして書き込む。そして処理は終了する。

【００７５】

このような作動によって、例えば図７で示されたブロックを含むモデル３と、「Ｖ６」を示す選択情報４を入力してコード生成ツール２を作動させると、コー

ド生成ツール２のステップ８１０～８６０の処理によって、図９に示すような中間モデルが生成される。そしてステップ８７０によってこの中間モデルに対応するソースコードが生成される。したがって、この生成されたソースコードにはＶ８用、Ｉ６用に特化されたコードは含まれることはない。

【００７６】

なお、ステップ８１０～８６０の処理は、図２に示した生成モデル抽出エンジン２１の機能を実現し、ステップ８７０の処理は、コード生成エンジン２２の機能を実現する。

【００７７】

以上のような構成および作動のパーソナルコンピュータ１、コード生成ツール２の効果について以下説明する。

【００７８】

Ｖ６、Ｖ８、Ｉ６等、複数のバリエーションに対応するモデルを作成し、ソースコードの作成時に個々のバリエーションにのみ向けられたソースコードを作成したい場合、それぞれのバリエーションに対応する部分をサブシステムとし、そのサブシステムの属性情報中に、使用バリエーションの項目を設け、その項目の値を、当該サブシステムが対応するバリエーションとする。そして、ソースコード作成時に、選択情報４として特定のバリエーションの名称をファイルまたはユーザによる操作によってコード生成ツール２に入力し、また当該モデル３をコード生成ツール２に入力する。なお、使用バリエーションの項目の値としては、Ｖ６、Ｖ８、Ｉ６のいずれか１つであってもよいし、あるいはそれらのうちの任意の２つ、あるいは３つ全部を含む値であってもよい。例えば、Ｖ６、Ｉ６に共通に用いられるサブシステムは、使用バリエーションの項目としてＶ６、Ｉ６の２つの値を含むようにする。また、全バリエーションで共通して用いられるサブシステムについては、使用バリエーションの項目としてＶ６、Ｖ８、Ｉ８の全ての値を含むようにすればよい。このように、サブシステムの属性情報中の使用バリエーションの項目値が、部分指定子に対応する。

【００７９】

これによって、コード生成ツール２は選択情報４に示された名称を使用バリエ

ーションの項目の値に含まないサブシステムをモデル３から削除した上で、ソースコードを生成する。したがって、複数のバリエーションに対応するモデルからソースコードが生成される開発環境において、当該ソースコードから不要なバリエーションに対応するソースコードが除かれていることになる。ひいては、このソースコードからコンパイル、リンクによって生成されるプログラムサイズの低減、およびこのプログラムを記録するメモリの容量の圧迫の緩和が実現される。

【００８０】

また、複数のバリエーションに対応するモデルからさらにモデルが生成されるプログラム開発環境において、生成されるモデルから当初のモデルの不要な部分に対応するモデルが除かれているようにすることも可能となる。

【００８１】

なお、別の例として、自己の属性情報中に使用バリエーションの項目を有さないサブシステム、または使用バリエーションの項目の値がＶ６、Ｖ８、Ｉ６のいずれをも含んでいないサブシステムについては、使用バリエーションの項目としてＶ６、Ｖ８、Ｉ８の全ての値を含むとみなしてステップ８４０の処理を行ってもよい。これによって、全てのバリエーションに共通に用いられるサブシステムについては、その属性情報中に能動的にその旨を記述する必要がなくなる。

【００８２】

（第３実施形態）

以下、本発明の第３実施形態について説明する。なお、本実施形態において第２実施形態と同一の部分があれば、その説明は省略または簡略化する。

【００８３】

図１０に、本実施形態におけるコード生成ツール２の構成、作動を概略的に示す。

【００８４】

本実施形態が第２実施形態と異なるのは、第１に、第２実施形態の部分指定子が属性情報中の使用バリエーションの項目値であったことに対し、本実施形態の部分指定子がサブシステムの名称となることである。そして第２に、選択情報４と部分指定子との相関関係を示す相関情報として切替マトリクス６が用いられる

ことである。

【００８５】

図１１に、本実施形態においてコード生成ツール２に入力されるモデル３の一部を例示する。このモデル３は、Ａサブシステム８１、Ｂサブシステム８２、Ｃサブシステム８３、Ｄサブシステム８４、およびＥサブシステム８５を有している。それぞれのサブシステムの名称の情報は、それぞれのサブシステムの属性情報に含まれている。

【００８６】

図１２に、切替マトリクス６の一例を示す。切替マトリクス６は、モデル３に含まれる全サブシステムを行項目として有し、Ｖ６、Ｖ８、Ｉ６といったバリエーションを列項目として有する行列を示す情報である、切替マトリクス６の各セルには１つのフラグが割り当てられている。図１２においては、フラグがセットされているセルを丸印で示し、フラグがセットされていない、すなわちリセットされているセルを空白で示している。各フラグのセット、リセットは、あらかじめユーザが設定できる。このように、切替マトリクス６は行項目のバリエーションと列項目のサブシステムとの相関関係を表している。

【００８７】

図１３に、本実施形態のコード生成ツール２において、ユーザが所定の操作をすることによって開始される、モデルからソースコードを生成する処理を示す。

【００８８】

ステップ９１０、９２０の処理は図８に示したステップ８１０、８２０の処理と同等である。

【００８９】

そしてステップ９２３では、ユーザによって指定された切替マトリクス６を読み込む。具体的には、切替マトリクス６として保存されているＨＤＤ１５中のファイルを読み出す。更に読み出した切替マトリクス６の、ステップ９２０で読み出した選択情報４が示すバリエーションに相当する列項目中で、フラグがセットされているセルを検索し、該当する各セルが存在する行のサブシステムのリストを選択リストとしてＲＡＭ１３に記録する。

【００９０】

次にステップ９２５では、図８のステップ８２５と同等の処理でモデル３中のサブシステムの検索を行う。

【００９１】

次にステップ９３０で、サブシステムリストの先頭にあるサブシステムの名称の情報を読み出し、その後当該サブシステムの情報をサブシステムリストから削除する。そしてステップ９４０で、読み出した属性情報中のサブシステムの名称が、選択リストと一致するか否かを判定する。一致しなければ処理はステップ９５０に進み、ＲＡＭ１３中のモデル３を、モデル３から当該サブシステムが削除されたものに書き換え、その後ステップ９６０に進む。ステップ９４０で一致すれば、処理は直接ステップ９６０に進む。

【００９２】

ステップ９６０では、全てのサブシステムの名称について読み込みが完了したか否かを判定する。具体的な判定処理は図８のステップ８６０と同等である。否定と判定すれば、処理はステップ９３０に戻る。肯定の場合、処理はステップ９７０に進み、ＲＡＭ１３に記録されているモデル３から、生成ルール２３に従ってソースコードを生成し、生成されたソースコードをＨＤＤ１５中にファイルとして書き込む。そして処理は終了する。

【００９３】

このような作動によって、例えば図１１で示されたブロックを含むモデル３と、「Ｖ６」を示す選択情報４と、図１２のようにセルをセットした切替マトリクス６とを入力してコード生成ツール２を作動させると、コード生成ツール２のステップ９１０～９６０の処理によって、図１４に示すような中間モデルが生成される。そしてステップ９７０によってこの中間モデルに対応するソースコードが生成される。したがって、この生成されたソースコードにはＶ８用、Ｉ６用に特化されたコードは含まれることはない。

【００９４】

なお、ステップ９１０～９６０の処理は、図１０に示した生成モデル抽出エンジン２１の機能を実現し、ステップ９７０の処理は、コード生成エンジン２２の

機能を実現する。

【００９５】

以上のような構成および作動のパーソナルコンピュータ１、コード生成ツール２の効果について以下説明する。

【００９６】

Ｖ６、Ｖ８、Ｉ６等、複数のバリエーションに対応するモデルを作成し、ソースコードの作成時に個々のバリエーションにのみ向けられたソースコードを作成したい場合、それぞれのバリエーションに対応する部分をサブシステムとし、それらのサブシステムの名称とバリエーションとの相関関係を示す切替マトリクス６を作成する。そして、ソースコード作成時に、選択情報４として特定のバリエーションの名称をファイルまたはユーザによる操作によってコード生成ツール２に入力し、また作成された切替マトリクス６をコード生成ツール２に入力し、また当該モデル３をコード生成ツール２に入力する。

【００９７】

これによって、コード生成ツール２は選択情報４に示されたバリエーションおよび切替マトリクス６のセットされたフラグによって相関づけられたサブシステム以外のサブシステムをモデル３から削除した上で、ソースコードを生成する。したがって、複数のバリエーションに対応するモデルからソースコードが生成される開発環境において、当該ソースコードから不要なバリエーションに対応するソースコードが除かれていることになる。ひいては、このソースコードからコンパイル、リンクによって生成されるプログラムサイズの低減、およびこのプログラムを記録するメモリの容量の圧迫の緩和が実現される。

【００９８】

また、複数のバリエーションに対応するモデルからさらにモデルが生成されるプログラム開発環境において、生成されるモデルから当初のモデルの不要な部分に対応するモデルが除かれているようにすることも可能となる。

【００９９】

また、第１および第２実施形態と異なり、モデル３の各サブシステムは自己の名称のみを有していればよいので、モデル３がバリエーションの切り替えのため

の情報を特別に有する必要がなく、切替マトリクス６と選択情報４のみで切り替えのための情報を一元管理することができる。

【０１００】

（第４実施形態）

図１５に、本発明の第４実施形態に係るシミュレーションツール７の構成、作動を概略的に示す。

【０１０１】

シミュレーションツール７は、入力されたモデル３からソースコードを生成しないまま、パーソナルコンピュータ１上でそのモデル３によって表現されるプログラムを実行するプログラムである。これは、モデル３からコード生成ツール２によってソースコードを生成し、そのソースコードから作成されたオブジェクトコードをエンジンＥＣＵ等に実装する前に、モデル３に基づくプログラムの動作の試験、確認を行うためのものである。

【０１０２】

シミュレーションツール７は、その機能面からシミュレーションモデル抽出エンジン２４およびシミュレーションエンジン２５に分けることができる。

【０１０３】

シミュレーションモデル抽出エンジン２４は、第３実施形態における生成モデル抽出エンジン２１と同等の機能を有する。具体的には、モデル３、選択情報４、切替マトリクス６の入力に対して、図１３のステップ９１０～９６０に示したような処理を行い、中間モデルを生成する。

【０１０４】

シミュレーションエンジン２５は、シミュレーションモデル抽出エンジン２４が生成した中間モデルによって表現されたプログラムの機能を、パーソナルコンピュータ１上で実行する。具体的には、シミュレーションエンジン２５は、当該中間モデルを読み込み、その中間モデルをソースコードに変換することなく、それに表現された機能を実行する。そしてその実行の際、ユーザの選択による任意のブロックの入力データ、出力データ、およびそれらの相関関係等の情報をディスプレイ１１あるいはプリンタ等に出力する。データの相関関係の情報としては

、例えばアクセル開度と燃料噴射量との関係を示すグラフ等がある。

【０１０５】

なお、シミュレーションモデル抽出エンジン２４とシミュレーションエンジン２５との区別は、あくまでもコード生成ツール２の機能面から見た区別であって、必ずしもプログラムとしては分離している必要はなく、一体不可分のプログラムとして実現されていてもよい。

【０１０６】

モデル３、選択情報４、切替マトリクス６等については、第３実施形態と同様である。ただし、シミュレーションエンジン２５が当該モデルとそのモデルが実装されるエンジンＥＣＵ等のハードウェアとの間の信号の入出力を擬似的に実現するために、モデル３にはハードウェアからの入力信号を代替するブロックが付加されている。この付加されたブロックは、第３実施形態のコード生成ツール２に当該モデルを入力する際には取り除く必要がある。

【０１０７】

このようなシミュレーションツール７によって、複数のバリエーションに対応するモデルを実行して動作を確認するプログラム開発環境において、実行されるモデルを、当初のモデルの不要な部分に対応する機能を除いた形で実行することが可能となる。

【０１０８】

このような場合、モデル３は、実際にはエンジンＥＣＵ等の特定のハードウェア上のみで実行されるプログラムとして開発されるとしても、プログラムの動作の試験、確認については、当該ハードウェアにプログラムを実装する前に行うことができるので、開発の利便性が向上する。

【０１０９】

（他の実施形態）

上記した各実施形態においては、Ｖ６、Ｖ８、Ｉ６といった、生成されるプログラムが対象とする機種についてのバリエーションを例示し、選択情報４はこの対象とする機種の情報を含むようになっていた。しかし、必ずしも選択情報４が含むバリエーションは生成されるプログラムが対象とする機種についてのもので

ある必要はなく、例えば日本向け、米国向け、ヨーロッパ向けといった仕向けについてのものであってもよいし、または試験用、量産用といったプログラムの使用目的についてのものであってもよい。仕向けによってプログラムの機能が異なる例としては、例えば各国内法のエンジンに対する規制が異なるため、それぞれの法規制に対応する部分が異なっている場合がある。

【０１１０】

図１６に、仕向けによって異なるブロックの構成例を示す。（ａ）は米国用に作成されたサブシステム中の一部のブロック構成であり、（ｂ）は米国以外用に作成されたサブシステム中の、（ａ）に対応する部分である。ブロック９１は、米国用、米国以外用の両方のサブシステムが共に有するブロックであり、２つの入力データに対して所定の処理を施して１つのデータを出力する。フィルタブロック９２は、ブロック９１からの入力データのうち、所定範囲内の値のデータのみそのまま出力し、それ以外のデータは出力しない機能を有するブロックである。米国では例えば排出ガス中の特定の成分を抑えるために、ブロック９１の出力データの上限と下限を法規制により限定しているとする。フィルタブロック９２は、この法規制の限定を実現するよう設定される。

【０１１１】

また、第４実施形態においては、シミュレーションモデル抽出エンジン２４は図１３のステップ９１０～９６０の処理によって実現できるとなっているが、必ずしもこれらの処理のように、不要なサブシステムの除去に切替マトリクス６を利用する必要はない。例えば、図８のステップ８１０～８６０の処理によって実現することもできる。また、図５のステップ５１０～５６０の処理によっても実現することができる。これらの場合には、図１０に示した切替マトリクス６は不要である。

【０１１２】

また、第１～第４実施形態のコード生成ツール２およびシミュレーションツール７は、その処理の過程において中間コードを生成しているが、必ずしもこのようになっている必要はなく、コード生成ツール２においてはモデル３から直接ソースコードを生成し、シミュレーションツール７においてはモデル３を直接実行

すてもよい。ただし、その際のソースコードからは、モデル３の不要な部分が除かれており、またシミュレーションツール７の実行はモデル３の不要な部分に対応する機能を除いた形で実行される必要がある。

【０１１３】

なお、本発明の第１～３実施形態においては、パーソナルコンピュータ１がコード生成装置および中間モデル生成装置を構成する。

【０１１４】

また、第４実施形態においては、パーソナルコンピュータ１がシミュレーション装置および中間モデル生成装置を構成する。

【０１１５】

また、第１～３実施形態においては、コード生成ツール２がコード生成プログラムおよび中間モデル生成プログラムを構成する。

【０１１６】

また、第４実施形態においては、シミュレーションツール７がシミュレーションプログラムおよび中間モデル生成プログラムを構成する。

【０１１７】

また、図５のステップ５１０、図８のステップ８１０、図１３のステップ９１０のそれぞれの処理が、モデル取得手段を構成する。

【０１１８】

また、図５のステップ５２０、図８のステップ８２０、図１３のステップ９２０のそれぞれの処理が、選択情報取得手段を構成する。

【０１１９】

また、図５のステップ５３０～５７０、図８のステップ８２５～８７０、図１３のステップ９２５～９７０のそれぞれの処理が、部分指定子および選択情報に基づいて、モデルから部分指定子が指定する部分を取り除いたものからソースコードを生成する除去生成手段を構成する。

【０１２０】

また、図５のステップ５３０～５６０、図８のステップ８２５～８６０、図１３のステップ９２５～９６０のそれぞれの処理が、部分指定子および選択情報に

基づいて、モデルから部分指定子が指定する部分を取り除いたモデルを生成する除去生成手段を構成する。

【０１２１】

また、シミュレーションモデル抽出エンジン２４とシミュレーションエンジン２５が、部分指定子および選択情報に基づいて、モデルから部分指定子が指定する部分を取り除いたモデルを生成する除去生成手段を構成する。

【０１２２】

また、図１３のステップ９２３が、部分指定子および選択情報との相関関係を示す相関情報を取得する相関情報取得手段を構成する。

【図面の簡単な説明】

【図１】

本発明の第１実施形態に係るコード生成装置としてのパーソナルコンピュータ１の構成を示す図である。

【図２】

コード生成ツール２等の構成、作動を概略的に示す図である。

【図３】

コード生成ツール２に入力されるモデル３の一例を示す図である。

【図４】

エンジンの各気筒の状態を加算して出力するブロックの図である。

【図５】

コード生成ツール２の、モデルからソースコードを生成する処理のフローチャートである。

【図６】

図３のモデル３から生成される中間モデルの一部の図である。

【図７】

第２実施形態のコード生成ツール２に入力されるモデル３の一例を示す図である。

【図８】

第２実施形態のコード生成ツール２の、モデルからソースコードを生成する処

理のフローチャートである。

【図９】

図７のモデル３から生成される中間モデルの一部の図である。

【図１０】

第３実施形態のコード生成ツール２等の構成、作動を概略的に示す図である。

【図１１】

第３実施形態のコード生成ツール２に入力されるモデル３の一例を示す図である。

【図１２】

切替マトリクス６の一例を示す図である。

【図１３】

第３実施形態のコード生成ツール２の、モデルからソースコードを生成する処理のフローチャートである。

【図１４】

図１１のモデル３から生成される中間モデルの一部の図である。

【図１５】

第４実施形態に係るシミュレーションツール７の構成、作動を概略的に示す図である。

【図１６】

仕向けによって異なるブロックの構成例を示す図である。

【図１７】

従来の切り替え方法で表現したモデルの一部を示す図である。

【図１８】

図１７に示した部分に対応するソースコードを示す図である。

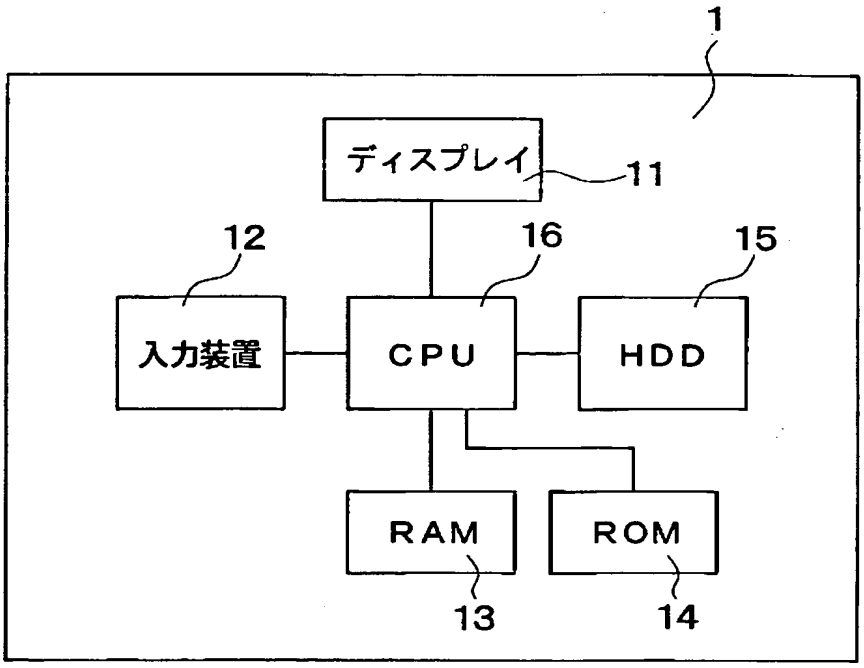
【符号の説明】

１…パーソナルコンピュータ、２…コード生成ツール、３…モデル、
４…選択情報、５…コンパイラ、リンカ、６…切替マトリクス、
７…シミュレーションツール、１１…ディスプレイ、１２…入力装置、
１３…ＲＡＭ、１４…ＲＯＭ、１５…ＨＤＤ、１６…ＣＰＵ、

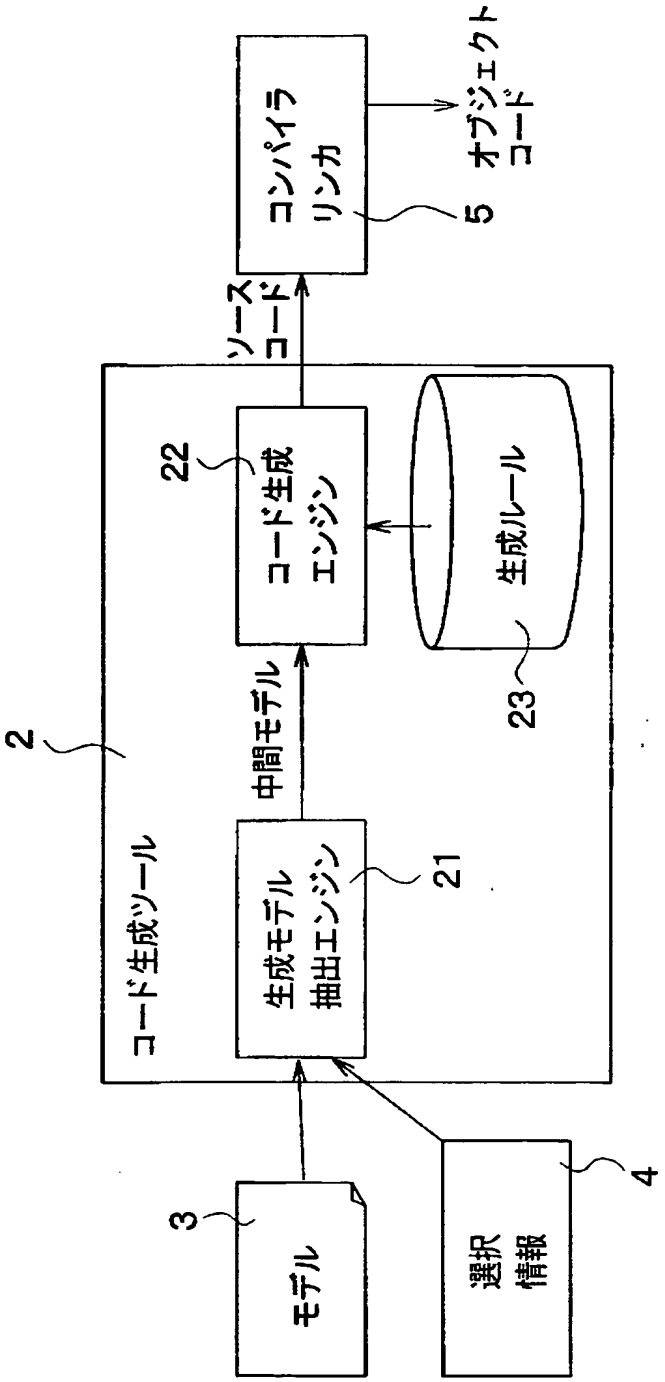
２１…生成モデル抽出エンジン、２２…コード生成エンジン、
２３…生成ルール、２４…シミュレーションモデル抽出エンジン、
２５…シミュレーションエンジン、３１…Ｖ６スタート、
３２…Ｖ６用サブシステム、３３…Ｖ６エンド、３４…Ｖ８スタート、
３５…Ｖ８用サブシステム、３６…Ｖ８エンド、３７…Ｉ６スタート、
３８…Ｉ６用サブシステム、３９…Ｉ６エンド、５１…Ｖ６用サブシステム、
５２…Ｖ８用サブシステム、５３…Ｉ６用サブシステム、
５４…選択ブロック、５５…入力端子。

【書類名】　　図面

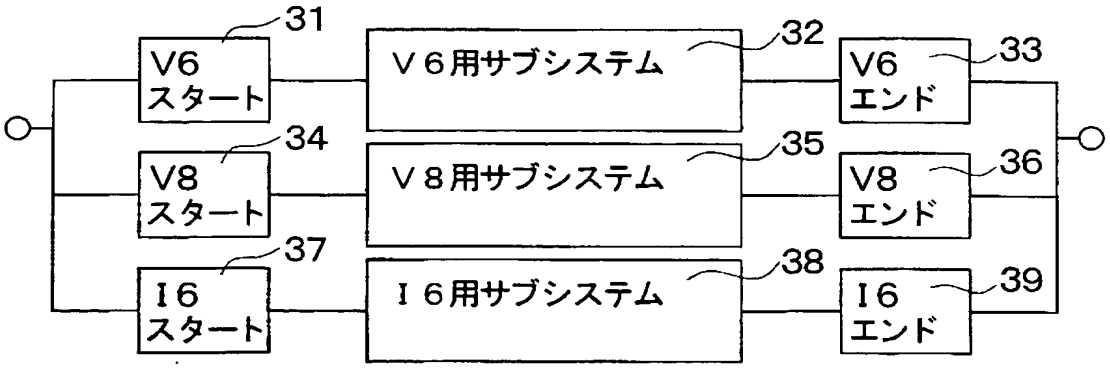
【図１】



【図２】

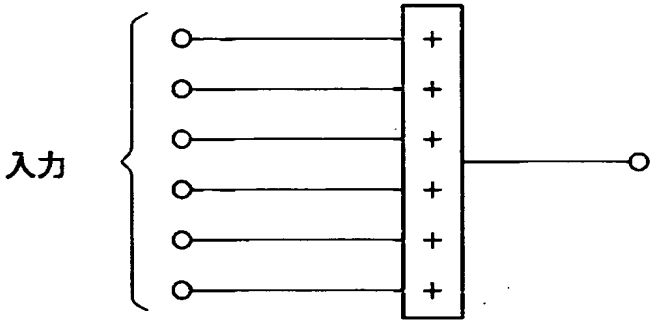

コンパイラ
リンカ
5
オブジェクト
コード
ソース
コード
22
コード生成
エンジン
生成ルール
23
2
中間モデル
コード生成ツール
生成モデル
抽出エンジン
21
3
モデル
選択
情報
4

【図３】

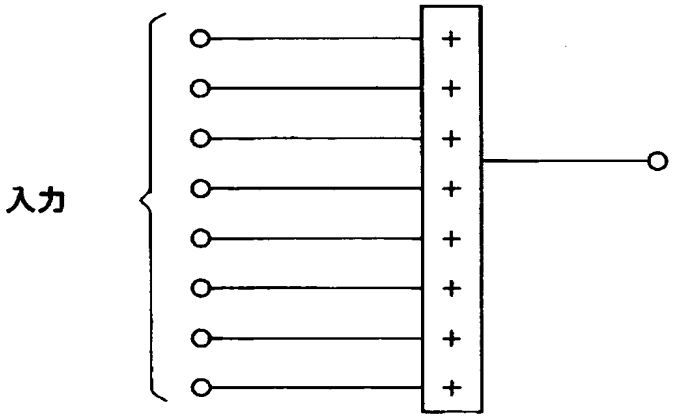


【図４】

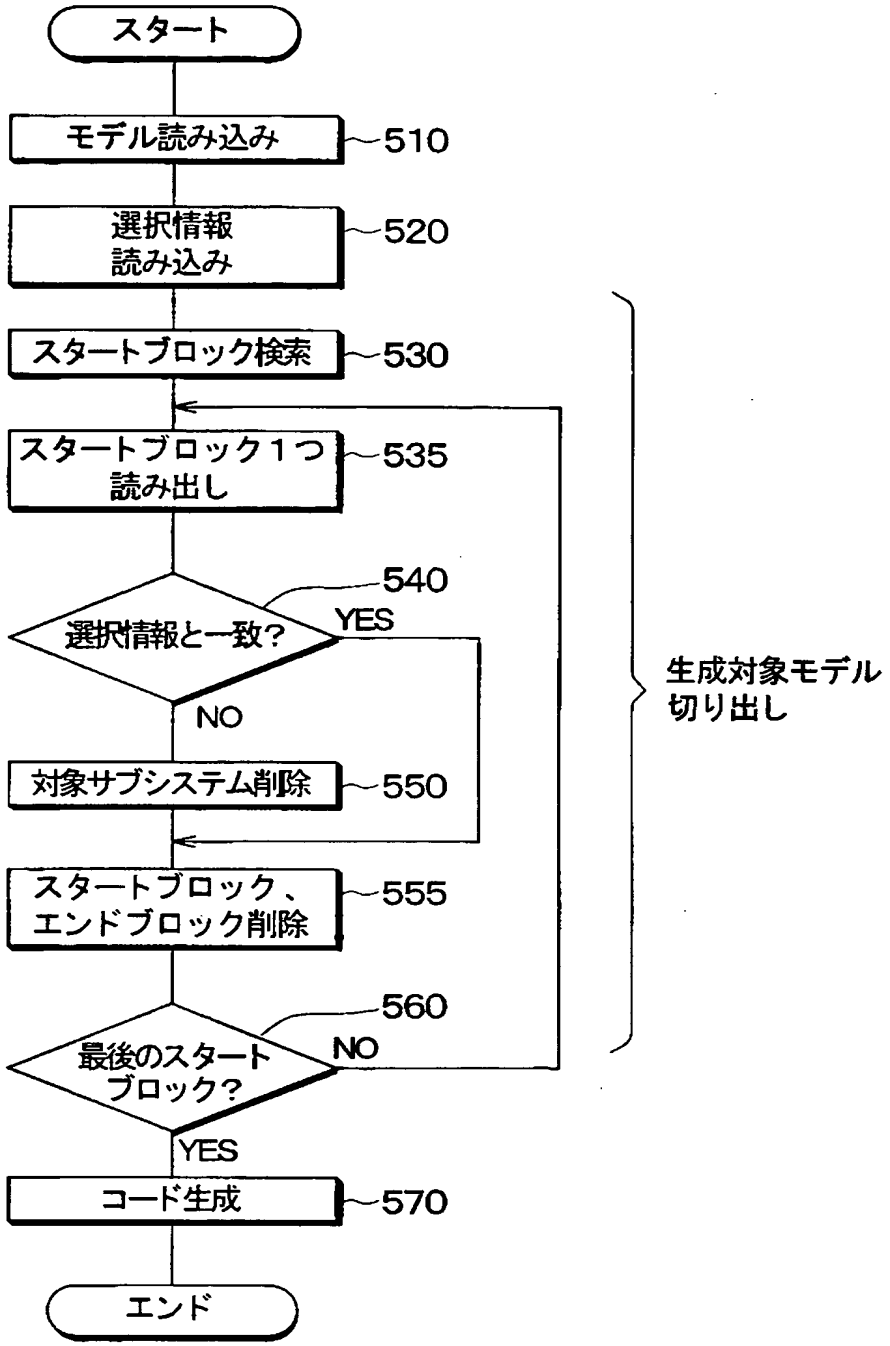
（ａ）６気筒用サブシステム



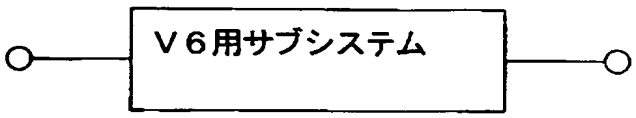
（ｂ）８気筒用サブシステム



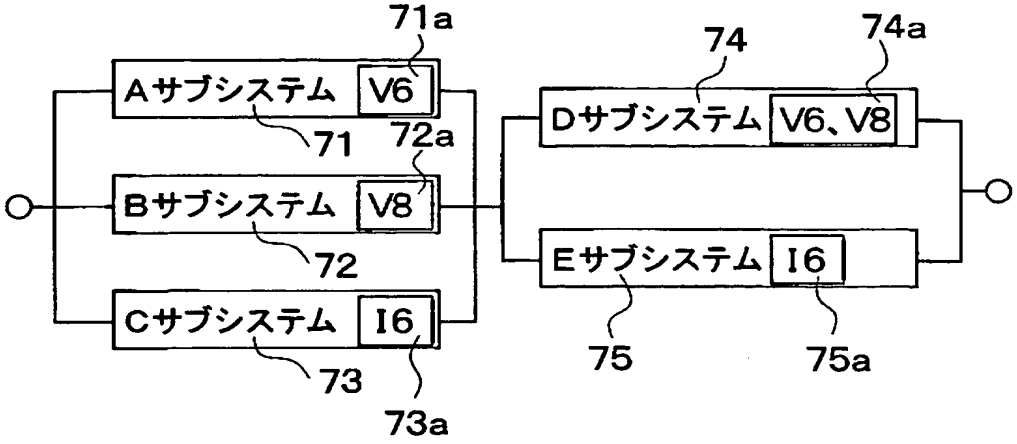
【図５】


スタート
モデル読み込み
510
選択情報
読み込み
520
スタートブロック検索
530
スタートブロック１つ
読み出し
535
540
選択情報と一致?
YES
NO
対象サブシステム削除
550
スタートブロック、
エンドブロック削除
555
560
最後のスタート
ブロック?
NO
YES
コード生成
570
エンド
生成対象モデル
切り出し

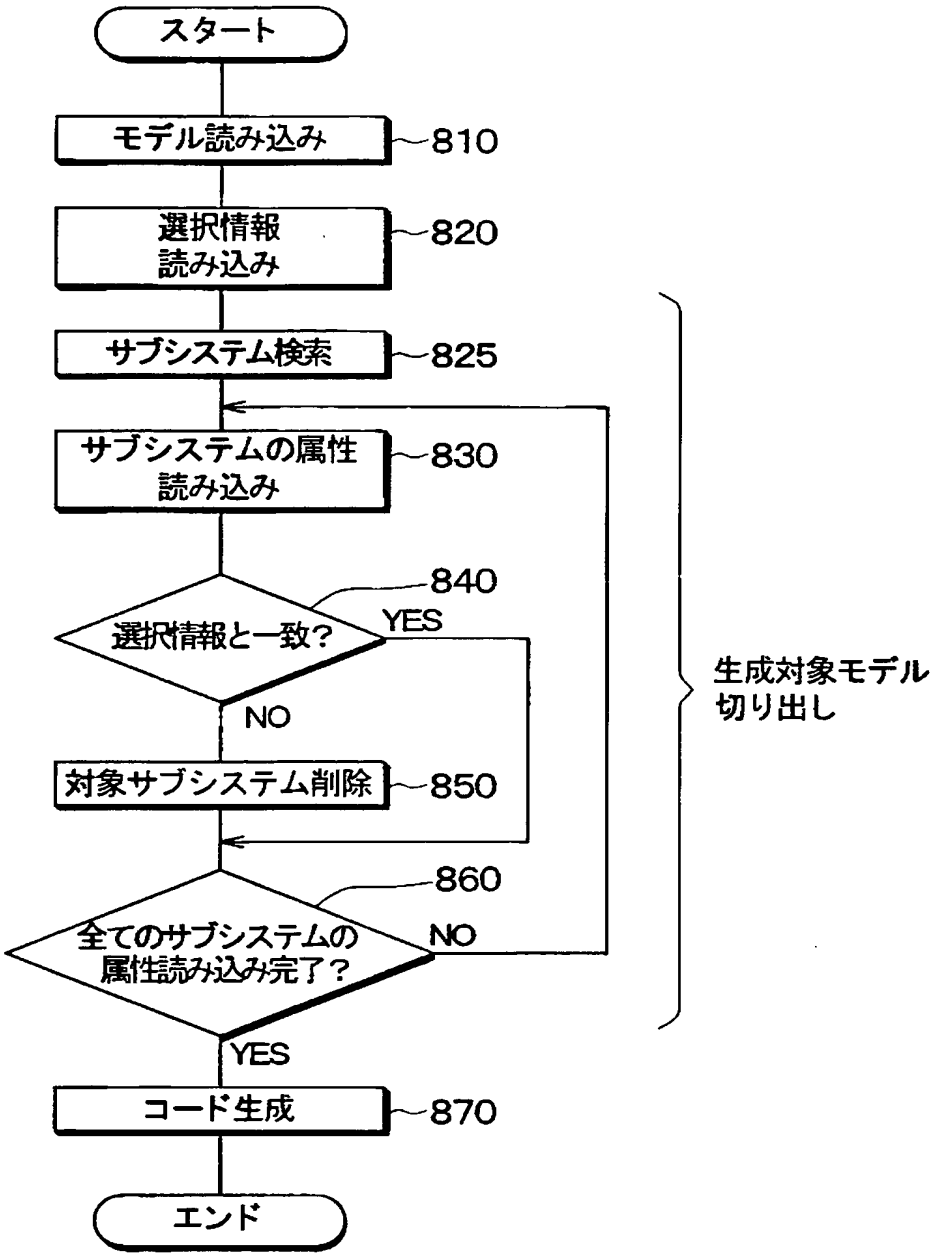
【図６】


Ｖ６用サブシステム

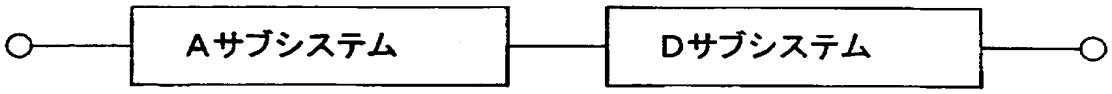
【図７】


71a
Aサブシステム
V6
71
72a
Bサブシステム
V8
72
Cサブシステム
I6
73
73a
74
74a
Dサブシステム
V6、V8
Eサブシステム
I6
75
75a

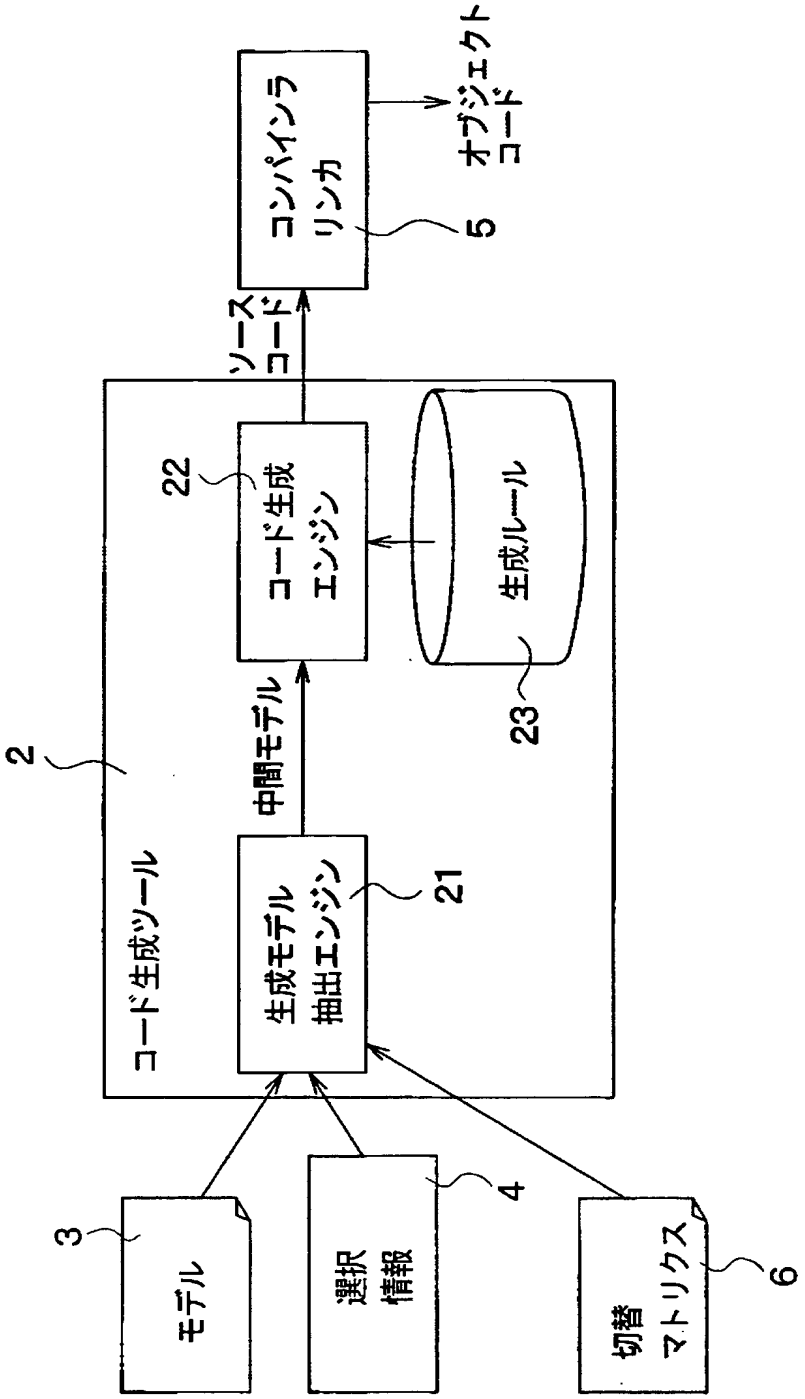
【図８】


スタート
モデル読み込み
810
選択情報
読み込み
820
サブシステム検索
825
サブシステムの属性
読み込み
830
840
選択情報と一致?
YES
NO
対象サブシステム削除
850
860
全てのサブシステムの
属性読み込み完了?
NO
YES
コード生成
870
エンド
生成対象モデル
切り出し

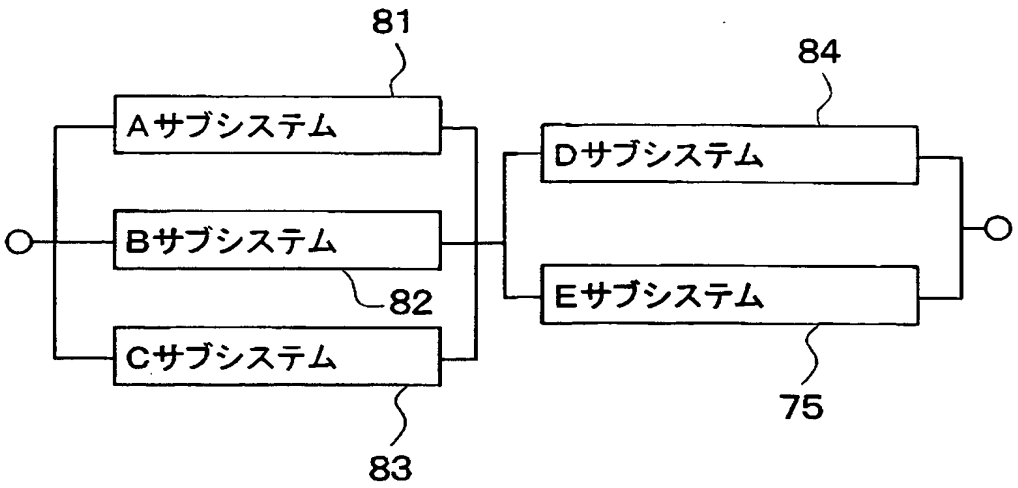
【図９】


Aサブシステム
Dサブシステム

【図１０】


コンパインラ
リンカ
5
オブジェクト
コード
ソース
コード
22
コード生成
エンジン
生成ルール
23
2
中間モデル
コード生成ツール
生成モデル
抽出エンジン
21
3
モデル
選択
情報
4
切替
マトリクス
6

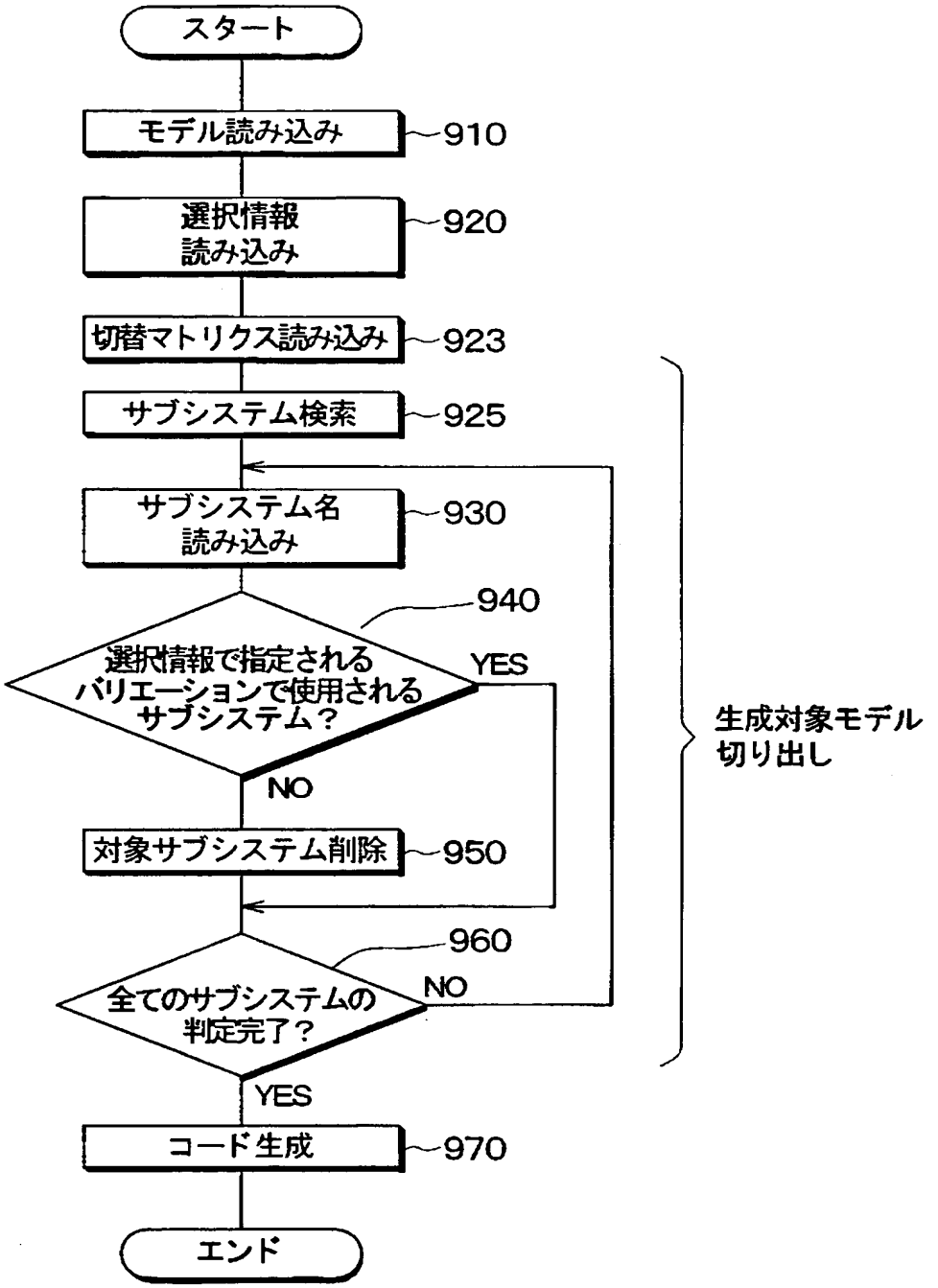
【図１１】



【図１２】

	V6	V8	I6
A サブシステム	○		
B サブシステム		○	
C サブシステム			○
D サブシステム	○	○	
E サブシステム			○

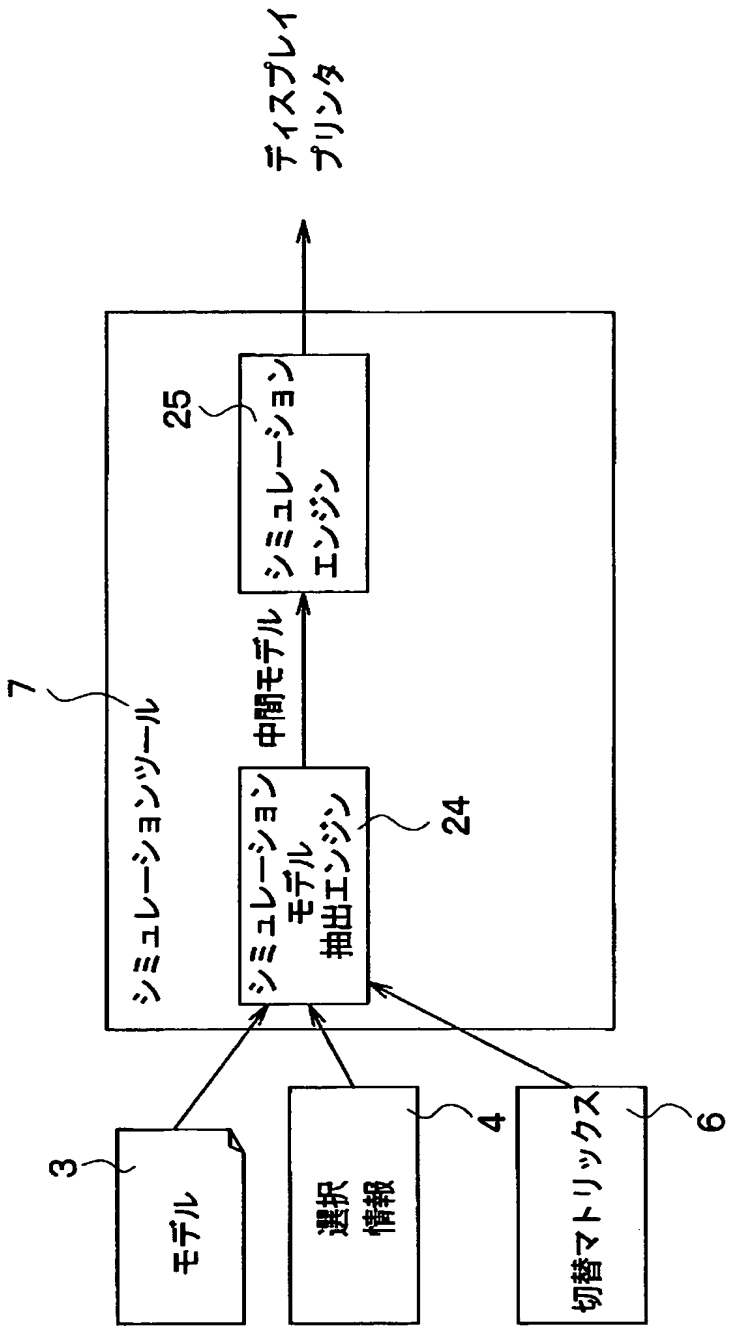
【図１３】


スタート
モデル読み込み
910
選択情報
読み込み
920
切替マトリクス読み込み
923
サブシステム検索
925
サブシステム名
読み込み
930
940
選択情報で指定される
バリエーションで使用される
サブシステム？
YES
NO
対象サブシステム削除
950
960
全てのサブシステムの
判定完了？
NO
YES
コード生成
970
エンド
生成対象モデル
切り出し

【図１４】

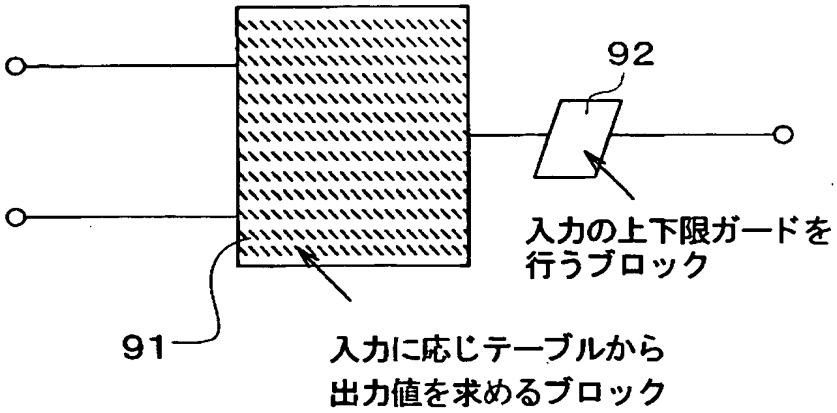

Aサブシステム
Dサブシステム

【図１５】

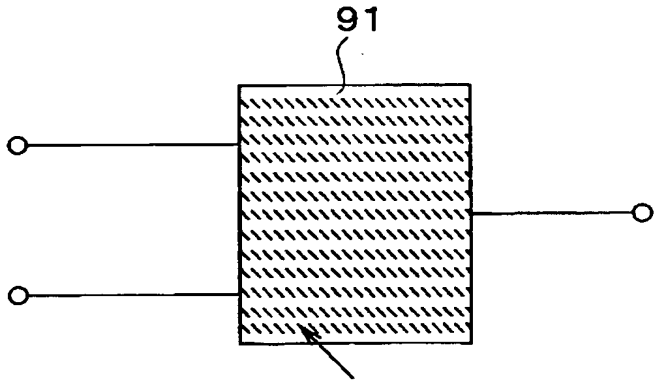

ディスプレイ
プリンタ
25
シミュレーション
エンジン
7
シミュレーションツール
中間モデル
シミュレーション
モデル
抽出エンジン
24
3
モデル
選択
情報
4
切替マトリックス
6

【図１６】

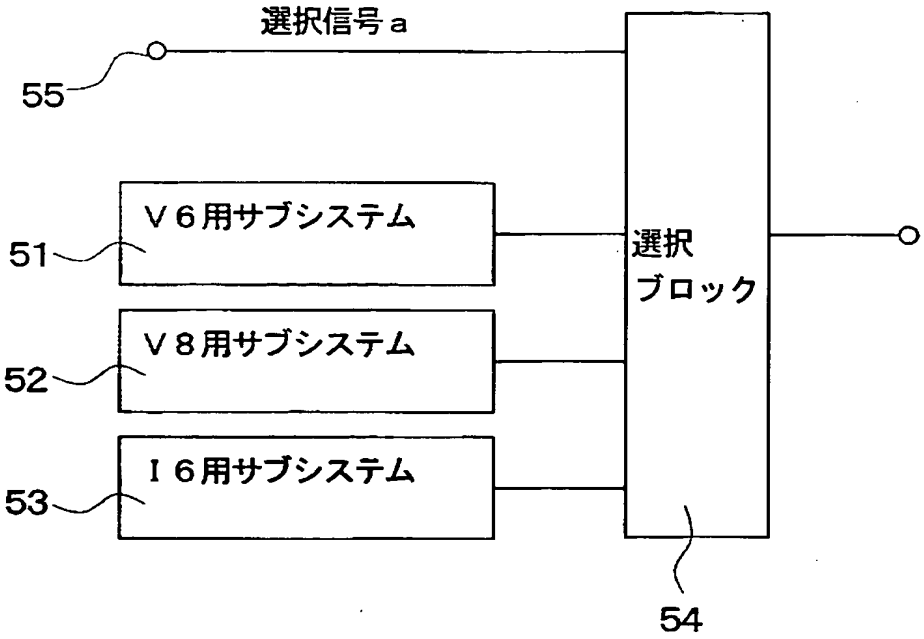
（ａ）米国用サブシステム



（ｂ）米国以外用サブシステム



【図１７】



【図１８】

```
Switch(a){
  case1:
   Ｖ６用サブシステム
   break;
  case2:
   Ｖ８用サブシステム
   break;
  case3:
   Ｉ６用サブシステム
   break;
}
```

【書類名】　　要約書

【要約】

【課題】　複数のバリエーションに対応するモデルからソースコードが生成される開発環境において、生成されるソースコードから当該モデルの不要な部分に対応するソースコードが除かれているようにすることが可能となる。

【解決手段】　複数のバリエーションに対応するモデルからソースコードを生成するコード生成ツール２を備えたパーソナルコンピュータが、ＨＤＤから自己の特定の部分を指定する部分指定ブロックを含むモデルを取得し、ＨＤＤまたは入力装置から、部分指定ブロックによって指定された部分の取捨選択に関する選択情報を取得し、この部分指定子および選択情報に基づいて、上記モデルから部分指定子が指定する部分を取り除いたものからソースコードを生成する機能を備える。

【選択図】　　図５

特願２００３－０１７６６８

出　願　人　履　歴　情　報

識別番号　　［０００００４２６０］

１．変更年月日　　１９９６年１０月　８日
［変更理由］　　名称変更
住　所　　愛知県刈谷市昭和町１丁目１番地
氏　名　　株式会社デンソー